Distributed Scheduling Algorithms for High-Speed Switching Systems

†Shunyuan Ye, ‡Yanming Shen, †Shivendra Panwar
†Department of Electrical and Computer Engineering, Polytechnic Institute of NYU
‡School of Computer Science and Technology, Dalian University of Technology, China
e-mail: sye02@students.poly.edu, shen@dlut.edu.cn, panwar@catt.poly.edu

***Abstract*—Given the rapid increase in traffic, greater demands have been put on research in high-speed switching systems. Such systems have to simultaneously meet several constraints, e.g., high throughput, low delay and low complexity. This makes it challenging to design an efficient scheduling algorithm, and has consequently drawn considerable research interest. However, previous results either cannot provide a 100% throughput guarantee without a speedup, or require a complex centralized scheduler. In this paper, we design a *distributed* 100% throughput algorithm for crosspoint buffered switches, called DISQUO, with very limited message passing. We prove that DISQUO can achieve 100% throughput for any admissible Bernoulli traffic, with a low time complexity of $O(1)$ per port and a few bits message exchanging in every time slot. To the best of our knowledge, it is the first distributed algorithm that can provide a 100% throughput for a crosspoint buffered switch.**

I. Introduction

With the growing Internet traffic demand, there is an increasing interest in designing large-scale high-performance packet switches. There is also a growing need for high-speed switching in the backplane of multiprocessing high-performance computer architectures [1], [2], and in large data centers [3], [4].

A scheduling algorithm is needed to schedule packet transmissions in such a system. A good algorithm has to meet several requirements, e.g., high throughput, low delay, and low complexity. In order to achieve these requirements, such switches usually require centralized, sometimes complex, algorithms. The well-known *maximum weight matching (MWM)* algorithm [5], [6] can achieve 100% throughput for any admissible arrival traffic, but it is not practical to implement due to its high computational complexity ($O(N^3)$). Also, the MWM algorithm needs a centralized scheduler. This increases the implementation complexity and leads to communication overhead. A number of practical iterative algorithms have been proposed, such as iSLIP [7] and DRRM [8]. However, they cannot guarantee 100% throughput for general arrival traffic patterns.

Due to the memory speed limit, most current switches use *input queuing (IQ)* [6], [7], [9], [10] or *combined input and output queuing (CIOQ)* [11]. To address the high complexity of designing scheduling algorithms for input-queued switching architectures, the *crosspoint buffered switching* architecture has been proposed, which promises a simpler scheduling algorithms and better delay performance [12]–[14]. For a crosspoint buffered switch, each input maintains virtual output queues (VOQs), one for each output, and each crosspoint contains a finite buffer. With a speedup of 2, the authors in [15], [16] showed that a crosspoint buffered switch can provide 100% throughput under any admissible traffic. However, without speedup, previous 100% throughput results are only limited to uniform traffic loads. Under uniform traffic, it has been shown that longest queue first at the input port and round-robin at the output port (LQF-RR) [13], or a simple round-robin scheduler at both input and output ports (RR-RR) [12], guaranteed 100% throughput. In [17], the authors proposed a distributed scheduling algorithm and derived a relationship between throughput and the size of crosspoint buffers. However, to achieve 100% throughput, an infinite-size crosspoint buffer was needed. To our knowledge, there is no distributed algorithm that can achieve 100% throughput for a *finite* crosspoint buffer without speedup.

There has always been a close relationship between the field of switch scheduling and scheduling transmissions in wireless ad hoc networks [5], [18]. Recently, it has been shown that CSMA-like algorithms [19]–[24] can achieve the maximum throughput in wireless ad hoc networks. Stations only need to sense the channel and make their scheduling decisions based on local queue information. These algorithms are easy to implement since no message passing is required. They are the first *distributed* algorithms that can achieve the maximum theoretical capacity in wireless networks.

Inspired by the CSMA-like algorithms and using many of the technique pioneered by them, we propose a distributed algorithm for crosspoint buffered switches that can stabilize the system under any admissible Bernoulli traffic. Note that for such CSMA-like algorithms to work properly, a node has to know its neighbors state in the previous slot by carrier sensing. This can be achieved in a wireless network due to the broadcast property of the medium. However, this cannot be easily implemented in a switching system. We make a key observation that the crosspoint buffers can be used for implicit message passing. By observing the buffer states, an input can get some partial information on whether the corresponding outputs are busy or not. This requires no change in the switch fabric architecture or implementation. Based on this observation, we designed DISQUO; the DIStributed QUeue input-Output scheduler. With DISQUO, an input only uses its local queue information and the locally observable partial schedule in the previous time slot to make its scheduling decision. We prove the stability of the system and evaluate

the performance of DISQUO by running extensive simulations. For technical reasons, each input does need to have the global maximum queue size in the system, which requires some message exchanging in each time slot. However, the simulations we conducted show that without the explicit message passing, the algorithm can still stabilize the system for the traffic patterns that we tested. Therefore, we propose the fully distributed algorithm without the global maximum queue size information as a conjecture. This result fulfills the long sought conjectured promise of this architecture [12], [13], [15]–[17]. The simulation results also show that it can provide good delay performance, comparable to output-queued switches, under different types of traffic.

The rest of the paper is organized as follows. Some theoretical preliminaries are presented in the next section. We present DISQUO in Section III and prove the system stability. Simulation results are presented in Section IV. We give the proof of system convergence in Section D, and conclude the paper in Section V.

II. Preliminaries

In this section, we introduce the notation and preliminary results that we will use in the theoretical proof of our algorithms.

A. Glauber Dynamics

A sequence of random variables (X_0, X_1, $\cdots$) is a *Markov chain* with state space $\mathbf{\Omega}$ and transition matrix $\mathbf{P}$ if for all x, $y \in \mathbf{\Omega}$, all $n \geq 1$, and all events $\mathcal{H}_{n-1} = \cup_{s=0}^{n-1}\{X_s = x_s\}$, we have

$$P\{X_{n+1} = y | \{X_n = x\} \cup \mathcal{H}_{n-1}\} = P\{X_{n+1} = y | X_n = x\} = P(x, y)$$

The process can then be described as:

$$\boldsymbol{\mu}(\tau) = \boldsymbol{\mu}(\tau - 1)\mathbf{P} = \boldsymbol{\mu}(0)\mathbf{P}^{\tau},$$

where $\boldsymbol{\mu}(\tau)$ is the probability distribution of X_τ.

The Markov chain is *irreducible* if any state can reach any other state. If the system starts from any state X and it can return to the state within finite time with a probability 1, the Markov chain is *positive recurrent*. If the Markov chain is irreducible and positive recurrent, it has a unique stationary distribution $\boldsymbol{\pi}$ so that:

$$\lim_{\tau \to \infty} \boldsymbol{\mu}(\tau) = \boldsymbol{\pi}.$$

Let P^* denote the transition probability matrix for the reverse Markov chain $(\cdots, X_n, X_{n-1}, \cdots)$. If $P = P^*$, the Markov chain is called *time-reversible* [25].

A *graph* $G = (V, E)$ consists of a *vertex set* V and an *edge set* E, where the elements of E are unordered pair of vertices: $E \subset \{\{x, y\} : x, y \in V, x \neq y\}$. If $\{x, y\} \in E$, y is a *neighbor* of x (and also x is a neighbor of y). Let $\mathcal{N}(x)$ denote all the neighbors of x. A *independent set* $I \subset V$ is a set such that if $x \in I$, $\forall y \in \mathcal{N}(x)$, $y \notin I$. Let $\mathcal{I}(G)$ represent all the independent sets of G.

Definition 1. *Consider a graph $G(V, E)$, with $\mathbf{W} = [W_i]_{i \in V}$ a vector of weights associated with the vertices. Glauber dynamics [25] is a Markov chain over $\mathcal{I}(G)$. Suppose that the chain is at state $\mathbf{X}(n-1) = [X_i(n-1)]_{i \in V}$. The next transition of Glauber dynamics follows the rules:*

- *Select a vertex $i \in V$ uniformly at random.*
- *If $\forall j \in \mathcal{N}(i)$, $X_j(n-1) = 0$, then*

$$X_i(n) = \begin{cases} 1 & \text{with probability } \frac{\exp(W_i)}{1+\exp(W_i)} \\ 0 & \text{otherwise.} \end{cases}$$

- *Otherwise, $X_i(n) = 0$.*

The Glauber dynamics is irreducible, aperiodic and time-reversible over $\mathcal{I}(G)$ [25]. It has a product-form stationary distribution, which is:

$$\pi(\mathbf{X}) = \frac{1}{Z} \exp(\sum_{i \in \mathbf{X}} W_i); \mathbf{X} \in \mathcal{I}(G), \tag{1}$$

where Z is a normalizing constant in order to have the sum of the probabilities unit total mass.

B. Mixing Time

Glauber dynamics can converge to its stationary distribution starting from any initial distribution. To characterize the convergence speed, we need to quantify the time that it takes for Glauber dynamics to reach close to its stationary distribution. To establish the result, we need to define some notation first.

Definition 2. *(Distance of probability distributions) Given two probability distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ over a finite space $\mathbf{\Omega}$, the total variance (TV) distance is defined as:*

$$||\boldsymbol{\mu} - \boldsymbol{\nu}||_{TV} := \frac{1}{2} \sum_{i \in \mathbf{\Omega}} |\mu(i) - \nu(i)|, \tag{2}$$

and the χ^2 distance [26], represented as $\|\frac{\boldsymbol{\mu}}{\boldsymbol{\nu}} - 1\|_{2,\boldsymbol{\mu}}$, is defined as:

$$\left\|\frac{\boldsymbol{\nu}}{\boldsymbol{\mu}} - 1\right\|_{2,\boldsymbol{\mu}}^2 := \left\|\boldsymbol{\nu} - \boldsymbol{\mu}\right\|_{2,\frac{1}{\boldsymbol{\mu}}}^2 = \sum_{i \in \mathbf{\Omega}} \mu(i) \Big(\frac{\nu(i)}{\mu(i)} - 1\Big)^2. \tag{3}$$

For any two vectors, $\boldsymbol{\mu}$, $\boldsymbol{\nu} \in \mathbb{R}_+^{|\mathbf{\Omega}|}$, we define:

$$\|\boldsymbol{\nu}\|_{2,\boldsymbol{\mu}}^2 := \sum_{i \in \mathbf{\Omega}} \mu_i \nu_i^2. \tag{4}$$

Following the definition, the probability distances satisfy the following condition.

Lemma 1. *[26] Given two probability distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ over a finite space $\mathbf{\Omega}$, the equation below holds:*

$$\left\|\frac{\boldsymbol{\nu}}{\boldsymbol{\mu}} - 1\right\|_{2,\boldsymbol{\mu}} \geq 2||\boldsymbol{\nu} - \boldsymbol{\mu}||_{TV} \tag{5}$$

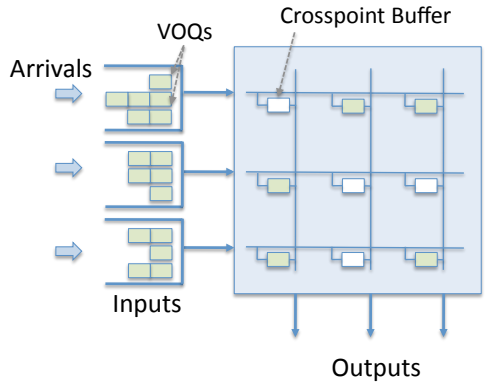

Fig. 1. An example of a crosspoint buffered switch. Each input has N *virtual output queues* (VOQs). There is a buffer with a size of K at each crosspoint of the fabric.

Proof:

$$\begin{aligned}\left\|\frac{\boldsymbol{\nu}}{\boldsymbol{\mu}}-1\right\|_{2,\boldsymbol{\mu}} &= \sqrt{\sum_{i\in\mathbf{\Omega}}\mu(i)\Big(\frac{\nu(i)}{\mu(i)}-1\Big)^2}\\ &= \sqrt{\sum_{i\in\mathbf{\Omega}}\mu(i)}\sqrt{\sum_{i\in\mathbf{\Omega}}\mu(i)\Big(\frac{\nu(i)}{\mu(i)}-1\Big)^2}\\ &\geq \sum_{i\in\mathbf{\Omega}}\mu(i)\Big|\frac{\nu(i)}{\mu(i)}-1\Big|\\ &\quad \text{(using Cauchy-Schwarz inequality)}\\ &= \sum_{i\in\mathbf{\Omega}}|\nu(i)-\mu(i)| = 2||\boldsymbol{\nu}-\boldsymbol{\mu}||_{TV} \qquad (6)\end{aligned}$$

■

Definition 3. *[25] (Mixing time) For a Markov chain with a transition probability matrix* $\mathbf{P}$ *and a stationary distribution* $\boldsymbol{\pi}$*, define the distance:*

$$d(\tau) := \max_{\boldsymbol{\mu}(0)} \|\boldsymbol{\mu}(0)\mathbf{P}^{\tau}-\boldsymbol{\pi}\|_{TV}. \qquad (7)$$

The mixing time is defined as:

$$\tau_{mix}(\delta) := \min\{\tau : d(\tau)\leq\delta\}. \qquad (8)$$

From the definition, we can see that mixing time is a parameter to measure the convergence rate of a Markov chain to its stationary distribution. Also, following Lemma 1, the mixing time can be measured by calculating the χ^2 distance of $\boldsymbol{\mu}(\tau)$ and $\boldsymbol{\pi}$.

III. DISQUO: A Distributed Algorithm for a Crosspoint Buffered Switch

In this section, we will present DISQUO for a crosspoint buffered switch. The algorithm is totally distributed. Inputs and outputs utilize the states of crosspoint buffers to implicitly exchange information. We will prove the system stability for any admissible Bernoulli traffic, and evaluate the delay performance by running extensive simulations for different traffic patterns.

A. Crosspoint Buffered Switch

With today's ASIC technology, it is now possible to add a small buffer at each crosspoint inside the crossbar (see Fig. 1). This makes the *crosspoint buffered* or *combined input and crossbar queueing* (CICQ) switch a much more attractive architecture since its scheduler is potentially much simpler. The input and output schedulers can be independent. First, each input picks a crosspoint buffer to send a packet to. Then, each output picks a crosspoint buffer to transmit a packet from. However, existing algorithms [12], [13], [27], [28] either cannot guarantee 100% throughput or require a centralized scheduler.

An $N\times N$ crosspoint buffered switch is shown in Fig. 1. We assume fixed size packet (cell) switching. Variable size packets can be segmented into cells before switching and reassembled at the output ports. There are *virtual output queues (VOQs)* at the inputs to prevent head-of-line blocking. Each input maintains N VOQs, one for each output. Let VOQ_{ij} represent the VOQ at input i for output j, and $Q_{ij}(n)$ the queue length of VOQ_{ij} at time n. Let (i,j) represent the crosspoint between input i and output j.

Each crosspoint has a buffer of size K. Most current implementations are constrained by the buffer size. However, it turns out that $K=1$ is sufficient for DISQUO. We will therefore assume that $K=1$ in the following. Our algorithm can be easily extended to the case when $K>1$. Let CB_{ij} denote the buffer at the crosspoint between input i and output j. $B_{ij}(n)\in\{0,1\}$ denotes the occupancy of CB_{ij} at time n.

A schedule can be represented by $\mathbf{S}(n) = [\mathbf{S}^I(n),\mathbf{S}^O(n)]$. $\mathbf{S}^I(n) = [S^I_{ij}(n)]$ is the input schedule. Each input port can only transmit at most one cell at each time slot. Thus the input schedule is subject to the following constraints:

$$\sum_j S^I_{ij}(n)\leq 1,\ S^I_{ij}(n)=0 \text{ if } B_{ij}(n)=1. \qquad (9)$$

$\mathbf{S}^O(n) = [S^O_{ij}(n)]$ is the output schedule. It has to satisfy the following constraints:

$$\sum_i S^O_{ij}(n)\leq 1,\ S^O_{ij}(n)=0 \text{ if } B_{ij}(n)=0. \qquad (10)$$

Let λ_{ij} represent the arrival rate of traffic between input i and output j. We assume that the arrival process is i.i.d. Bernoulli.

Definition 4. *An arrival process is said to be* admissible *if it satisfies:*

$$\sum_j \lambda_{ij}<1, \text{ and } \sum_i \lambda_{ij}<1. \qquad (11)$$

Let $\boldsymbol{\sigma}^*$ denote the traffic that the equivalence in Eq. (11) holds. It is easy to verify that for any admissible traffic $\boldsymbol{\sigma}$, there exists an $\epsilon>0$ such that $\boldsymbol{\sigma}<(1-\epsilon)\boldsymbol{\sigma}^*$ component-wise.

Let $||\mathbf{Q}||$ represent the norm of matrix $\mathbf{Q}$: $||\mathbf{Q}|| = \left(\sum_{i,j}Q^2_{ij}\right)^{1/2}$. The stability of a system is defined as:

Definition 5. *A system of queues is said to be* stable *if:*

$$\lim_{n\to\infty}\sup E||\mathbf{Q}(n)||<\infty. \qquad (12)$$

Theorem 1. *A scheduling algorithm, which can stabilize the system for any admissible traffic in a bufferless crossbar switch, can also stabilize the system for any admissible traffic in a crosspoint buffered switch [28].*

Proof: Please refer to Property 1 of Ref. [28]. ■

Following Theorem 1, all the scheduling algorithms that have been proposed for an input-queued switch, e.g., the *maximum weight matching* (MWM) [5], can be applied to a crosspoint buffered switch. As we will show later, the reason that DISQUO can stabilize the system for any admissible traffic is that, after the system converges, the schedule generated at every time slot has a weight that approaches the one with the maximum weight matching schedule.

B. DISQUO Scheduling Algorithms

Before presenting the algorithm, we need to introduce some further notation. A DISQUO schedule $\mathbf{X}(n)$ is a schedule that is generated by the DISQUO algorithm. It is used to determine the input schedules and output schedules. A DISQUO schedule has the following properties:

Property 1. *A DISQUO schedule $\mathbf{X}(n)$ can be represented by an $N \times N$ matrix, where $X_{ij}(n) \in \{0, 1\}$, and $\sum_i X_{ij}(n) \leq 1$, $\sum_j X_{ij}(n) \leq 1$.*

With some abuse of notation, we also use $\mathbf{X}$ to represent a set, and write $(i, j) \in \mathbf{X}$ if $X_{ij} = 1$. Note that a DISQUO schedule $\mathbf{X}$ has the property that if $X_{ij} = 1$, then $\forall i' \neq i$, $X_{i'j} = 0$ and $\forall j' \neq j$, $X_{ij'} = 0$. We define these crosspoints as its *neighbors* as follows.

Definition 6. *The neighbors of a crosspoint (i, j) are defined as:*

$$\mathcal{N}(i, j) = \{(i', j) \text{ or } (i, j') \mid \forall i' \neq i, \ \forall j' \neq j\} \tag{13}$$

A DISQUO schedule $\mathbf{X}$ then has the following property:

Property 2. *If $(i, j) \in \mathbf{X}$, $\forall (k, l) \in \mathcal{N}(i, j)$, $(k, l) \notin \mathbf{X}$.*

Let $\mathcal{X}$ represent the set of all DISQUO schedules.

Property 3. *At each time slot, when a DISQUO schedule is generated, each input and output port determine their schedules by observing the following rules:*

- *For input i, when $X_{ij}(n) = 1$, if $Q_{ij}(n) > 0$ and $B_{ij}(n-1) = 0$, $S^I_{ij}(n) = 1$. Otherwise, $S^I_{ij}(n) = 0$.*
- *For output j, if $X_{ij}(n) = 1$ and $B_{ij}(n) > 0$, $S^O_{ij}(n) = 1$.*

Property 4. *For an input i, if $\forall j$, $X_{ij} = 0$, then it is referred to as a* free input. *A free input port has the freedom to pick any eligible crosspoint to serve, i.e. it can transfer a packet to any empty crosspoint buffer.*

Property 5. *For an output port j, if $\forall i$, $X_{ij} = 0$, then it is refered a* free output. *A free output is free to pick any non-empty crosspoint to serve.*

Following Prop. 3-5, the input schedule $\mathbf{S}^I(n)$ and output schedule $\mathbf{S}^O(n)$ can be determined after the DISQUO schedule $\mathbf{X}(n)$ is generated. Therefore, we will next present the basic DISQUO schedule updating algorithm that generates $\mathbf{X}(n)$.

At the beginning, set the initial DISQUO schedule $\mathbf{X}(0)$ to any schedule that satisfies Property 1. For simplicity, we can set $\mathbf{X}(0) = \emptyset$. At the beginning of a time slot n, generate an input/output permutation $\mathbf{H}(n)$ randomly. Then, the DISQUO schedule $\mathbf{X}(n)$ is updated following the rules below:

Basic DISQUO Algorithm

- ∀ $(i, j) \notin \mathbf{H}(n)$:
  - (a) $X_{ij}(n) = X_{ij}(n-1)$.
- For $(i, j) \in \mathbf{H}(n)$:
  - If $(i, j) \in \mathbf{X}(n-1)$:
    - (b) $X_{ij}(n) = 1$ with probability p_{ij};
    - (c) $X_{ij}(n) = 0$ with probability $\overline{p}_{ij} = 1 - p_{ij}$.
  - Else, if $(i, j) \notin \mathbf{X}(n-1)$, and $\forall (k, l) \in \mathcal{N}(i, j)$, $X_{kl}(n-1) = 0$, then:
    - (d) $X_{ij}(n) = 1$ with probability p_{ij};
    - (e) $X_{ij}(n) = 0$ with probability $\overline{p}_{ij} = 1 - p_{ij}$.
  - Else, $(i, j) \notin \mathbf{X}(n-1)$ and $\exists (k, l) \in \mathcal{N}(i, j)$ such that $X_{kl}(n-1) = 1$:
    - (f) $X_{ij}(n) = 0$.

---

p_{ij} is defined as: $p_{ij} = \frac{\exp(W_{ij}(n))}{1+\exp(W_{ij}(n))}$, where $W_{ij}(n)$ is a weight function of the queue size $Q_{ij}(n)$, which is defined as

$$W_{ij}(n) = f(\tilde{Q}_{ij}(n)). \tag{14}$$

$f(\cdot)$ is a concave function which we will define later, $Q_{max}(n) = \max_{i,j} Q_{ij}(n)$, and $\tilde{Q}_{ij}(n) = \max\{f^{-1}(\frac{\epsilon}{2N^2} f(Q_{max}(n))), Q_{ij}(n)\}$. Recall that for any admissible traffic $\boldsymbol{\sigma}$, there exists an $\epsilon > 0$ such that $\boldsymbol{\sigma} < (1-\epsilon)\boldsymbol{\sigma}^*$ component-wise. Thus, ϵ is a small positive number that satisfies the condition $\boldsymbol{\sigma} < (1-\epsilon)\boldsymbol{\sigma}^*$.

Note that in our algorithm, $X_{ij}(n)$ can change only when (i, j) is in $\mathbf{H}(n)$. Therefore, at every time slot, only $(i, j) \in \mathbf{H}(n)$ can join or leave the DISQUO schedule.

Comparing the algorithm with the updating rules of Glauber dynamics, we can see that, DISQUO schedule essentially is a multiple-update version of Glauber dynamics, with a vector of time-varying weights since the weight is a function of the queue length, which changes over time. At every time slot, instead of picking only one VOQ, multiple VOQs are picked by $\mathbf{H}(n)$ to update their states. Like the Glauber dynamic, $\mathbf{X}(n)$ also only depends on $\mathbf{X}(n-1)$. Thus, the DISQUO schedules $\mathbf{X}(0), \mathbf{X}(1), \cdots$ form a Markov chain. As we will show later, this Markov chain is irreducible, positive recurrent and time-reversible.

After DISQUO schedule $\mathbf{S}(n)$ is generated, inputs and outputs can update their schedules $\mathbf{S}^I(n)$ and $\mathbf{S}^O(n)$, and start the packet transmissions. As we will prove later, following this algorithm, the system is stable for any admissible Bernoulli i.i.d. traffic.

C. Discussions

As we presented in the previous section, the decision of making a crosspoint (i, j) active or not is based on a probability p_{ij}, which depends on not only the local queue sizes, but also a global information $Q_{max}(n)$. However, since $\frac{\epsilon}{2N^2}$ is very small, we can use $W_{ij}(n) = f(Q_{ij}(n))$ directly for real

implementation. The introduction of $Q_{max}(n)$ is primarily for technical reasons. Therefore, we have the following conjecture.

Conjecture 1. *DISQUO scheduling algorithm with the weight function defined as $W_{ij}(n) = f(Q_{ij}(n))$ can still stable the system for any admissible traffic.*

To be precise, we still need some message passing between linecards. As suggested in [19], [26], a rough estimate of $Q_{max}(n)$ is sufficient to guarantee the convergence of the system. Therefore, a low-rate Ethernet connection, which is typical in nowadays router design for backplane control message passing, can be used for linecards to broadcast their local maximum queue sizes so that other linecard can estimate the value of $Q_{max}(n)$. At time slot $kN + i$, only linecard i broadcasts its local maximum queue size. Since at every time slot, there is at most one packet departure/arrival from/to an input, the estimation of $Q_{max}(n)$, denoted as $\tilde{Q}_{max}(n)$, satisfies: $Q_{max}(n) - 2N \leq \tilde{Q}_{max}(n) \leq Q_{max}(n) + 2N$. This is sufficient for the system stability. For details, please refer to Ref. [19], [26].

Recall that we need to generate an input/output permutation $\mathbf{H}(n)$ *randomly* at every time slot to update the DISQUO schedule. For the simplicity of practical implementation, we can generate the schedule following a Hamiltonian walk [29], instead of using a totally random algorithm. For a switch of size N, there are $N!$ input/output permutations. The Hamiltonian walk schedule $\mathbf{H}(n)$ visits each of the $N!$ distinct permutations exactly once during every $N!$ slots in a deterministic periodic schedule. This can be simply implemented with a time complexity of $O(1)$ [29].

In the following, we will show how this conjecture can be implemented in a distributed way, without message passing. We will show the performance of the switch by running extensive simulations.

D. Distributed Implementation

As shown in the basic DISQUO scheduling algorithm, $\mathbf{X}(n)$ is generated based on $\mathbf{X}(n-1)$. Therefore, each input i needs to keep track of the DISQUO schedule in the previous slot, i.e. for which output j was $X_{ij}(n-1) = 1$. Similarly, each output needs to keep track of for which input i was $X_{ij}(n-1) = 1$. Since the algorithm is distributed, there is no message passing between inputs and outputs. DISQUO needs to make sure that the inputs and outputs keep a consistent view of the DISQUO schedule. For example, if $X_{ij}(n) = 1$, both input i and output j should be aware of this.

Since the decision for a crosspoint to join or leave the DISQUO schedule needs the queue length information, inputs are responsible for making the decisions. However, there are two problems that have to be solved: 1) before input i decides to change X_{ij} from 0 to 1, it needs to check the states of all the neighbors of (i, j), namely, the status of output j, which is not directly accessible at input i; 2) after input i changes the value of X_{ij}, this information has to be passed over to output j. To solve the problems, the distributed DISQUO algorithm is designed to achieve *implicit* message passings by utilizing the crosspoint buffers.

The input and output scheduling algorithms work as follows.

Input Scheduling Algorithm (ISA)

At each input port i, assume $(i, j) \in \mathbf{H}(\mathbf{n})$.
1) $\forall j' \neq j$: (a) $X_{ij'}(n) = X_{ij'}(n-1)$.
2) ∘ If $X_{ij}(n-1) = 1$:
   (b) $X_{ij}(n) = 1$ with probability p_{ij};
   (c) $X_{ij}(n) = 0$ with probability $\overline{p}_{ij} = 1 - p_{ij}$.
   ∘ Else, if $X_{ij}(n-1) = 0$ and there exists a j' such that $X_{ij'}(n-1) = 1$, $X_{ij'}(n) = 1$ according to case (a) above. Consequently:
   (d) $X_{ij}(n) = 0$.
   ∘ Else, if there is no j' such that $X_{ij'}(n-1) = 1$, then input i was a free input:
   - If CB_{ij} is empty, output j was free:
     (e) $X_{ij}(n) = 1$ with probability p_{ij};
     (f) $X_{ij}(n) = 0$ with probability $\overline{p}_{ij} = 1 - p_{ij}$.
   - Else,
     (g) $X_{ij}(n) = 0$.
3) If $X_{ij}(n) = 1$, $Q_{ij}(n) > 0$ and $B_{ij}(n) = 0$, $S^I_{ij}(n) = 1$. Input i sends a packet to CB_{ij}. Otherwise, if input i is free, it generates $\mathbf{H}(n+1)$. Suppose that $(i, j') \in \mathbf{H}(n+1)$. If $CB_{ij'}$ is empty, input i serves it. Otherwise, it sends a packet to any empty crosspoint buffer except CB_{ij}.

Output Scheduling Algorithm (OSA)

For each output port j, assume (i, j) is selected by $\mathbf{H}(n)$.
1) $\forall i' \neq i$: (a) $X_{i'j}(n) = X_{i'j}(n-1)$.
2) ∘ If $X_{ij}(n-1) = 1$:
   (b) If at time n, CB_{ij} receives a packet from input i, $X_{ij}(n) = 1$.
   (c) Otherwise, $X_{ij}(n) = 0$.
   ∘ Else, if $X_{ij}(n-1) = 0$ and there exists a i' such that $X_{i'j}(n-1) = 1$, $X_{i'j}(n) = 1$. So:
   (d) $X_{ij}(n) = 0$.
   ∘ Else, there is no i' such that $X_{i'j}(n-1) = 1$, output j was free:
   - If input i sends a packet to CB_{ij} at the beginning of time n:
     (e) $X_{ij}(n) = 1$.
   - Else,
     (f) $X_{ij}(n) = 0$.
3) If $X_{ij}(n) = 1$, $S^O_{ij}(n) = 1$. Output j transmits a packet from CB_{ij}. Otherwise, output j is free, it generates $\mathbf{H}(n+1)$. Suppose that $(i', j) \in \mathbf{H}(n+1)$. If $CB_{i'j}$ is non-empty, output j serves it. Otherwise, output j picks any non-empty crosspoint to serve.

For the input scheduling algorithm, before input i decides to change X_{ij} from 0 to 1, it has to check the status of output j. But this information is not known at input i. If input i was free at time $n-1$, it has to generate $\mathbf{H}(n)$. Suppose that $(i, j) \in \mathbf{H}(n)$. Input i then transmits a packet to CB_{ij} if CB_{ij} was empty at time $n-1$. As we show in the output scheduling algorithm below, if output j was free, it transmits a packet from CB_{ij} at time $n-1$. Therefore, for case (e) and (f), input i can infer whether output j was free or not by

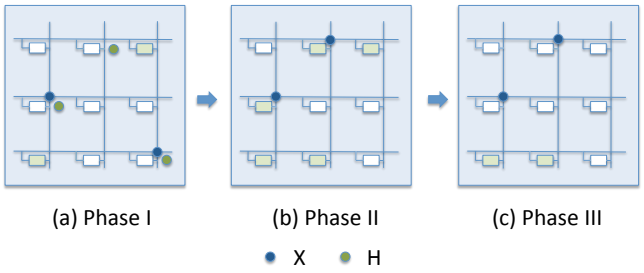

Fig. 2. An example of DISQUO schedule updating. One time slot is divided into three phases. $\mathbf{H}(n)$ is generated in Phase I to update $\mathbf{X}(n-1)$. In Phase II, inputs make their decisions and transmits packets to the crosspoint buffers. In Phase III, outputs updates their schedules and transmits packets from the crosspoint buffers. Only $(i,j) \in \mathbf{H}(n)$ can join or leave the DISQUO schedule.

observing the crosspoint buffer CB_{ij}. If the CB_{ij} is empty, output j was free at time $n-1$. Otherwise, output j was busy.

For the output schedulers, an output j has to observe the crosspoint buffer CB_{ij} to learn input i's decision following the output scheduling algorithm (OSA) above. For example, in case (b) and (c), it can learn input i's decision by observing whether a packet is sent to CB_{ij} or not at time n. Thus, when $(i,j) \in \mathbf{H}(n)$, a packet transmission from input i to CB_{ij} can implicitly pass its decision information to output j. If CB_{ij} is not empty at the beginning of time n, input i is not able to pass its decision information to output j. Recall that if output j was free, it can pick any non-empty crosspoint buffer to serve. Therefore, if output j was free at time $n-1$, it can calculate $\mathbf{H}(n)$ in advance, and transmit the packet from CB_{ij}, if $(i,j) \in \mathbf{H}(n)$ and CB_{ij} is not empty. By doing that, when $(i,j) \in \mathbf{H}(n)$, CB_{ij} will always be empty at the beginning of time n if output j was free, so that input i can pass its decision information to output j by sending a packet to the buffer. With this rule, when $(i,j) \in \mathbf{H}(n)$, input i can also directly infer that output j was busy if CB_{ij} is not empty at the beginning of time n, since otherwise, output j would have already transmitted that packet from CB_{ij} at time $n-1$.

E. Example

To help understand DISQUO, we give an illustrative example here. We assume that a schedule over one time slot can be divided into three phases: a) Phase I: every input and output calculate the Hamiltonian walk schedule $\mathbf{H}(n)$; b) Phase II: inputs update the DISQUO schedule based on $\mathbf{H}(n)$, and decide the value of $\mathbf{S}^I(n)$, after which packets can be sent from inputs to the crosspoint buffers; c) Phase III: outputs update the DISQUO schedule and decide the value of $\mathbf{S}^O(n)$ so that they can transmit packets from the crosspoint buffers.

As we can see from Fig. 2(a), the DISQUO schedule at time $n-1$ is $\mathbf{S}(n-1) = \{(2,1),(3,3)\}$. In Phase I, $\mathbf{H}(n) = \{(1,2),(2,1),(3,3)\}$ is generated at each input and output. In the following, we use the example to describe how a crosspoint joins or leaves the DISQUO schedule, and how the input/output scheduler $\mathbf{S}^I(n)$ and $\mathbf{S}^O(n)$ are decided after $\mathbf{X}(n)$ is generated.

- How a crosspoint joins the DISQUO schedule: $(1,2)$ is in $\mathbf{H}(n)$ and $X_{12}(n-1) = 0$. Also, input 1 knows that $\forall j$, $X_{1j}(n-1) = 0$ so that input 1 was a free input in the previous slot. If output 2 was also a free output, input 1 can decide whether to let $(1,2)$ join the DISQUO schedule or not, following case (e) or (f) of ISA. Input 1 cannot know the status of output 2 directly. However, it can learn output 2's status by observing CB_{12}. Since CB_{12} is empty, input i learns that output 2 was free. It can then decide whether to make $(1,2)$ active based on p_{12}. If its decision is to set $X_{12}(n)$ to 1, it should send a packet to CB_{12}. Otherwise, it remains a free input. In the example, the decision of input 1 is to set $X_{12}(n)$ to 1. Thus, $S^I_{12}(n) = 1$, and it sends a packet to CB_{12}, as shown in Fig. 2(b). Note that this transmission implicitly passes its decision information to output 2.
  Output 2 was a free output, and it observes that in Phase II, input 1 sends a packet to CB_{12}. Following case (e) of OSA, output 2 learns input 1's decision of setting $X_{12}(n)$ to 1. It then updates $X_{12}(n)$ to 1, and thus $S^O_{12}(n) = 1$. Output 2 transmits the packet from CB_{12}, which is shown in Fig. 2(c).
- How a crosspoint leaves the DISQUO schedule: $(3,3)$ is in $\mathbf{H}(n)$ and $X_{33}(n-1) = 1$. Following case (b) and (c), input 3 has to decide whether to keep $(3,3)$ in the DISQUO schedule or not, based on a probability p_{33} which is a function of the queue size Q_{33}. In the example, it decides to set $X_{33}(n)$ to 0. Input 3 becomes a free input. It calculates $\mathbf{H}(n+1)$, which we assume is $\{(1,3),(2,1),(3,2)\}$. Since $(3,2) \in \mathbf{H}(n+1)$ and CB_{32} is empty, it sets $S^I_{32}(n) = 1$, and sends a packet to CB_{32} (Fig. 2(b)). Note that by not sending a packet to CB_{33}, input 3 implicitly passes its decision of setting $X_{33}(n) = 0$ to output 3.
  Output 3 observes that, in Phase II, input 3 did not send any packet to CB_{33}. Following case (b), it learns input 3's decision and updates $X_{33}(n)$ to 0. Output 3 becomes a free output. Following the OSA, a free output has to generate $\mathbf{H}(n+1)$ at time n. Since $(1,3) \in \mathbf{H}(n+1)$ and CB_{13} is not empty, output 3 sets $S^O_{13}(n) = 1$ and transmits the packet from CB_{13}, as shown in Fig. 2(c).

From the example, we can see that, after $\mathbf{X}(n)$ is generated, which is $\{(1,2),(2,1)\}$, a packet is transmitted from input 1 to output 2, and one from input 2 to output 1. Besides that, input 3 and output 3, which are free, also transmit a packet. The transmissions by free inputs and outputs can be considered as an *augmentation* of $\mathbf{X}(n)$. In the following, we will show that the weight, only defined on $\mathbf{X}(n)$, is close enough to the maximum one to guarantee the throughput. The augmenting by free input/output, though it does not contribute to the stability of the switch, can improve the delay performance of the system.

F. Stationary Distribution

As mentioned before, $\{\mathbf{X}(n)\}$ forms a Markov chain. In this section, we will derive the stationary distribution of this Markov chain, and show that after the system converges, the weight of the DISQUO schedule is always very close to the MWM schedule in Lemma 9. We can then prove the system stability in Theorem 2.

Lemma 2. *If $\mathbf{X}(n-1) \in \mathcal{X}$, then $\mathbf{X}(n) \in \mathcal{X}$.*

Proof: If $\mathbf{X}$ is a DISQUO schedule it satisfies Property 1. For an input i, it is impossible that there exists $j \neq j'$ such that $X_{ij}(n) = X_{ij'}(n) = 1$, since before input i decides to change X_{ij} from 0 to 1, it always has to make sure that there does not exist a j' such that $X_{ij'}(n) = 1$.

For an output j, it is also impossible that there exists $i \neq i'$ such that $X_{ij}(n) = X_{i'j}(n) = 1$. This is because input i can change X_{ij} from 0 to 1 only when output j was free. So, $X_{ij}(n) = X_{i'j}(n) = 1$ only when input i and input i' decide to change the values from 0 to 1 at the same time slot, which requires both $(i,j) \in \mathbf{H}(n)$ and $(i',j) \in \mathbf{H}(n)$. But $\mathbf{H}(n)$ is an input/output permutation such that only one $(\cdot, j)$ is in $\mathbf{H}(n)$. Therefore, if $\mathbf{X}(n-1)$ satisfies Property 1, $\mathbf{X}(n)$ also satisfies Property 1. ∎

As mentioned before, $\mathbf{X}(n-1), \mathbf{X}(n), \cdots$ is a Markov chain since $\mathbf{X}(n)$ only depends on $\mathbf{X}(n-1)$. A transition from a state $\mathbf{X}$ to $\mathbf{X}'$ can occur only when the Hamiltonian walk schedule $\mathbf{H}(n)$ satisfies the condition:

$$(\mathbf{X} \cap \overline{\mathbf{X}'}) \cup (\overline{\mathbf{X}} \cap \mathbf{X}') \in \mathbf{H}(n).$$

This is because VOQs in $\mathbf{X} \cap \overline{\mathbf{X}'}$ leave the DISQUO schedule and $\overline{\mathbf{X}} \cap \mathbf{X}'$ join the DISQUO schedule. According to the DISQUO algorithm, only crosspoints in $\mathbf{H}(n)$ can join or leave the DISQUO schedule. Therefore, both $\mathbf{X} \cap \overline{\mathbf{X}'}$ and $\overline{\mathbf{X}} \cap \mathbf{X}'$ should be in $\mathbf{H}(n)$. The following lemma gives the transition probabilities.

Lemma 3. *If $\mathbf{X}$ can transit to $\mathbf{X}'$, the transition probability can be written as:*

$$p_n(\boldsymbol{X}, \boldsymbol{X}') = \sum_{\boldsymbol{H}: \mathbf{X} \triangle \mathbf{X}' \in \mathbf{H}} a(\boldsymbol{H}) \prod_{(i,j) \in \mathbf{X} \cap \overline{\mathbf{X}}'} \overline{p}_{ij} \prod_{(k,l) \in \overline{\mathbf{X}} \cap \mathbf{X}'} p_{kl} \cdot \prod_{(u,v) \in \mathbf{X} \cap \mathbf{X}' \cap \mathbf{H}} p_{uv} \prod_{(x,y) \in \mathbf{H} \cap \overline{\mathbf{X} \cup \mathbf{X}'} \cap \overline{\mathcal{N}(\mathbf{X} \cup \mathbf{X}')}} \overline{p}_{xy}, \tag{15}$$

where $a(\boldsymbol{H})$ is the probability that $\boldsymbol{H}$ is selected (which is $\frac{1}{N!}$), and $\mathbf{X} \triangle \mathbf{X}' = (\mathbf{X} \cap \overline{\mathbf{X}'}) \cup (\overline{\mathbf{X}} \cap \mathbf{X}')$.

Proof: Please refer to Appendix A. ∎

Lemma 4. *The Markov chain $\{\mathbf{X}(n)\}$ is irreducible and positive recurrent.*

Proof: Please refer to Appendix B. ∎

Since the Markov chain is positive recurrent, it has a unique stationary distribution. Let us associate each VOQ with a non-negative weight $W_{ij}(n) = f(Q_{ij}(n))$ at time n. The Markov chain has the following stationary distribution.

Lemma 5. *The Markov chain of the system has the following product-form stationary distribution:*

$$\pi_n(\mathbf{X}) = \frac{1}{\mathcal{Z}} \prod_{(i,j) \in \mathbf{X}} \frac{p_{ij}}{\overline{p}_{ij}} = \frac{1}{\mathcal{Z}} \prod_{(i,j) \in \mathbf{X}} e^{W_{ij}(n)}, \tag{16}$$

where

$$\mathcal{Z} = \sum_{\mathbf{X} \in \mathcal{X}} \prod_{(i,j) \in \mathbf{X}} \frac{p_{ij}}{\overline{p}_{ij}} = \sum_{\mathbf{X} \in \mathcal{X}} \prod_{(i,j) \in \mathbf{X}} e^{W_{ij}(n)}. \tag{17}$$

Proof: If a state $\mathbf{X}$ can make a transition to $\mathbf{X}'$, we can check that the distribution in Eq. (16) satisfies the detailed balance equation:

$$\pi_n(\mathbf{X}) p_n(\mathbf{X}, \mathbf{X}') = \pi_n(\mathbf{X}') p_n(\mathbf{X}', \mathbf{X}), \tag{18}$$

hence the Markov chain is reversible and Eq. (16) is the stationary distribution (see [30], Theorem 1.2). ∎

G. System Convergence

For Glauber dynamics, the weights are fixed over time. Therefore, the convergence rate of the system can be described by the distance between $\boldsymbol{\mu}$(n) and $\boldsymbol{\pi}$. However, following the algorithm presented in the previous section, the weights are changing over time such that the Glauber dynamics for each time slot n is different from those in other time slots, which means $\boldsymbol{\pi}_n$ also varies over time. To characterize the convergence rate of this system, we can define the distance

$$d_n = \|\boldsymbol{\mu}_n - \boldsymbol{\pi}_n\|_{TV}. \tag{19}$$

We aim to ensure that for any arbitrarily small $\delta > 0$, there exists a time $T_{mix}(\delta)$ that for any $n > T_{mix}(\delta)$, we have $d_n < \delta$ so that $\boldsymbol{\mu}_n$ and $\boldsymbol{\pi}_n$ are close enough. As compared to the definition of mixing time in Definition 3, $T_{mix}(\delta)$ shows the convergence rate of the system. Therefore, we will refer to $T_{mix}(\delta)$ as the mixing time of this inhomogeneous Markov chain.

In the following Lemma, we will prove that if the weight function $f(x)$, so that $W_{ij}(n) = f(Q_{ij}(n))$, is carefully selected, the system can always converge to the distribution $\boldsymbol{\pi}_n$ following DISQUO scheduling algorithm.

Lemma 6. *If $f(x) = \frac{\log(1+x)}{g(x)}$, then there exists a n^* that for any $\delta > 0$, $\|\boldsymbol{\mu}_n - \boldsymbol{\pi}_n\|_{TV} \leq \delta$ holds for all $n \geq n^*$, where $g(x)$ is a function that satisfies the following conditions:*

- *$g(x) \geq 1$, for all $x \geq 0$.*
- *$g'(x) \geq 0$, for all $x \geq 0$.*
- *$\lim_{x \to \infty} g(f^{-1}(x)) = \infty$.*

Proof: Please refer to Appendix D for the detailed proof. ∎

One example of $g(x)$ is $g(x) = \log(e + \log(1+x))$. Note that according to Lemma 6, if the weight function is well designed, the system will always converge to the product-form distribution as expressed in Eq. (16).

H. System Stability

As shown above, the Markov chain $\{\mathbf{X}(n)\}$ has a finite number of states and we already derived its stationary distribution. In the following, we will utilize MWM algorithm to prove the system stability. For an input-queued switch, the **MWM** algorithm selects a feasible schedule $\mathbf{S}(n)$ with the maximum weight:

$$\mathbf{S}^*(n) = \arg\max_{\mathbf{S} \in \mathcal{S}} \sum_{(i,j) \in \mathbf{S}} W_{ij}(n). \tag{20}$$

The algorithm can provide 100% throughput for any admissible traffic in a bufferless crossbar switch. According to

Theorem 1, MWM can be extended to a buffered crossbar switch. Following the DISQUO algorithm, if $X_{ij}(n) = 1$, and $Q_{ij}(n) > 0$ or $B_{ij}(n) > 0$, one packet can be transmitted from input i to output j. Therefore, we can define the weight of a DISQUO schedule as:

$$W(\mathbf{X}) = \sum_i \sum_j X_{ij}(n) W_{ij}(n). \tag{21}$$

For **MWM**, the result below has been established in [31].

Lemma 7. *For a scheduling algorithm, if given any ϵ and δ such that $0 \le \epsilon, \delta < 1$, there exists a $B > 0$ such that the scheduling algorithm satisfies the condition that in any time slot n, with a probability greater than $1 - \delta$, the scheduling algorithm can choose a feasible schedule $\mathbf{S}$ which satisfies the following condition:*

$$\sum_{(i,j)\in \mathbf{S}(n)} W_{ij}(n) \ge (1-\epsilon) \sum_{(k,l)\in \mathbf{S}^*(n)} W_{kl}(n), \tag{22}$$

whenever $||\boldsymbol{Q}(n)|| \ge B$, where $\boldsymbol{Q}(n) = [Q_{ij}(n)]_{i,j}$ and $||\boldsymbol{Q}(n)|| = \left(\sum_{i,j} Q_{ij}^2(n)\right)^{1/2}$. Then the scheduling algorithm can stabilize the system.

Theorem 2. *DISQUO can stabilize the system if the input traffic is admissible.*

Proof: Define the set:

$$\mathcal{K} = \{\mathbf{X} \in \mathcal{X} : W(\mathbf{X}) \le (1-\epsilon) W^*(\mathbf{X})\}.$$

According to Lemma 9 in Appendix C, for any $\delta > 0$, we have $\pi(\mathcal{K}) < \delta$, if the maximum weight satisfies the condition:

$$W^*(\mathbf{X}) > \frac{N^2 \log 2}{\epsilon\delta} > \frac{\log |\mathcal{X}|}{\epsilon\delta}. \tag{23}$$

So, for any $\epsilon, \delta > 0$, there exists a $B > N^3 \frac{\log 2}{\epsilon\delta}$ such that whenever $||\mathbf{Q}(n)|| > B$,

$$\sum_{i,j} Q_{ij}^2(n) > B^2 > N^6 \left(\frac{\log 2}{\epsilon\delta}\right)^2.$$

Then, $\max Q_{ij}^2(n) > N^4 \left(\frac{\log 2}{\epsilon\delta}\right)^2$. Thus, Eq. (23) holds and $\pi(\mathcal{K}) < \delta$. Hence the scheduling algorithm can stabilize the system according to Lemma 7. ■

IV. Simulations

In this section, we ran simulations for different scenarios to evaluate the performance of DISQUO. We also study the delay performance of the scheduling algorithm under different traffic patterns, including uniform and non-uniform traffic with Bernoulli and bursty arrivals. Note that DISQUO reduces to a heuristic scheduling algorithm for all arrival processes that are not i.i.d. Bernoulli. For bursty traffic, the burst length is distributed over $[1, 1000]$, following the truncated Pareto distribution:

$$P(l) = \frac{c}{l^\alpha}, \; l = 1, 2, \ldots, 1000, \tag{24}$$

where l is the burst length, α is the Pareto distribution parameter and c is the normalization constant. In the simulations, $\alpha = 1.7$, for which the average burst length is about 11.6. All inputs are equally loaded and we measure the packet delay. Simulations are run for long enough to ensure that the confidence intervals are small enough to make valid comparisons.

Fig. 3. Switch size N=32, uniform traffic for both Bernoulli i.i.d. and bursty arrivals

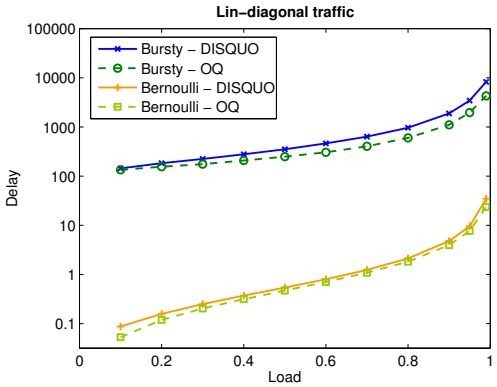

Fig. 4. Switch size N=32, lin-diagonal traffic for both Bernoulli i.i.d. and bursty arrivals

A. Uniform Traffic

For uniform traffic, a new cell is destined with equal probability to all output ports. Let σ represent the traffic load, and the arrival rate between input i and output j is $\sigma_{ij} = \frac{\sigma}{N}$. The delay performance of DISQUO under uniform Bernoulli and bursty traffic is shown in Fig. 3. We can see that the packet delay of DISQUO is very close to the output-queued switch (OQ). It has been shown that under uniform traffic, even an algorithm as simple as RR-RR can have a delay performance close to an output-queued switch [12]. However, the RR-RR algorithm cannot achieve 100% throughput when the traffic is non-uniform. Therefore, we will study the performance of DISQUO under non-uniform traffic next.

B. Non-uniform Traffic

We ran the simulations for the following non-uniform traffic patterns:

- Lin-diagonal: Arrival rates at the same input differ linearly, i.e, $\sigma_{i(i+j \pmod N)} - \sigma_{i(i+j+1 \pmod N)} = 2\sigma/N(N+1)$.

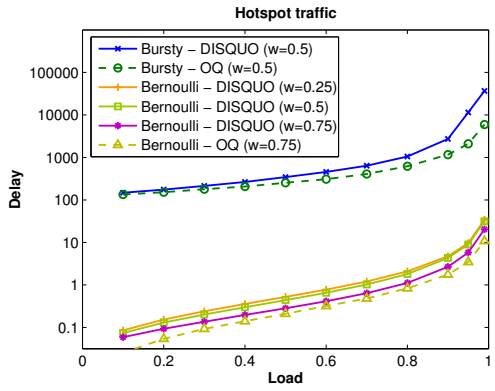

Fig. 5. Switch size N=32, hot-spot traffic for both Bernoulli i.i.d. and bursty arrivals

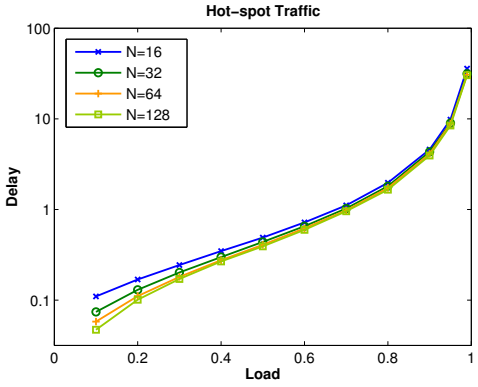

Fig. 6. Results of switches with different sizes, with hot-spot Bernoulli i.i.d. traffic, where $\omega = 0.5$

- Hot-spot: For input port i, $\sigma_{ii} = \omega\sigma$ and $\sigma_{ij} = (1 - \omega)\sigma/(N - 1)$, for $i \neq j$. We can get different traffic patterns by varying the hot-spot factor ω.

The delay performance for lin-diagonal and hot-spot traffic are shown in Fig. 4 and Fig. 5, respectively. We can see that under Bernoulli traffic, the delay performance of DISQUO is still very close to the output-queued switch. Packets have low delay even when the load is as high as 0.99. Note that the RR-RR algorithm can have a throughput of approximately only 85% [12] under hotspot traffic. Note that DISQUO is stable for the bursty traffic scenarios that we simulated.

C. Impact of Switch Size

We also study the impact of switch size on the delay performance. Generally, for input-queued switches, the average delay increases linearly with the switch size [7]. For output-queued switches, delay is independent of the size. Fig. 6 shows the delay performance of DISQUO with different switch sizes under Bernoulli hot-spot traffic, for which ω is 0.5. We can see that the delay is almost the same for different switch sizes. As the size increases, the delays even decrease slightly. This is partly because as the switch size increases, the number of crosspoint buffers increases as well, and the crosspoint buffers play a key role in reducing the average delay.

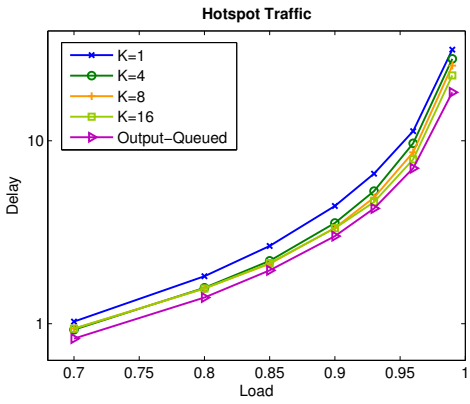

Fig. 7. Impact of buffer size, hot-spot traffic, $\omega = 0.5$, N=32

D. Impact of Buffer Size

If the buffer at each crosspoint increases to infinity, the buffered crossbar switch is then equivalent to an output-queued switch. So if we increase the buffer size, the average delay will decrease and converge to the delay of an output-queued switch. As we already showed in previous simulation results, the delay performance of DISQUO with a buffer size of 1 is already very close to that of an output-queued switch. Therefore, by increasing the buffer size, we can only get a very marginal improvement in delay performance. DISQUO can be easily modified for values of $K > 1$. Due to space considerations, we will not define DISQUO with $K > 1$ here. Fig. 7 shows the delay performance of DISQUO with different buffer sizes, under hot-spot traffic. We can see that the improvement is small. Therefore, we only need to implement a one-cell buffer at each crosspoint and still provide good delay performance. This is crucial since current technology limits the size of crosspoint buffers to a small number.

V. Conclusion

In this paper, we first proposed a distributed scheduling algorithm (DISQUO) for crosspoint buffered switches with a crosspoint buffer size of as small as one and no speedup. The computational complexity of DISQUO is only $O(1)$ per port, and we proved that it can achieve 100% throughput for any admissible Bernoulli i.i.d. traffic. We evaluated the performance of DISQUO by running extensive simulations. The results show that DISQUO can provide very good delay performance, as compared to an output-queued switch. With DISQUO, the average queuing delay for a packet is independent of the switch size, which makes it very suitable for large-scale switching system design.

References

[1] R. Hemenway, R. Grzybowski, C. Minkenberg, and R. Luijten, "Optical-packet-switched Interconnect for Supercomputer Applications," *Journal of Optical Networks*, vol. 3, no. 12, pp. 900–913, 2004.

[2] C. Minkenberg, F. Abel, R. Krishnamurthy, M. Gusat, P. Dill, I. Iliadis, R. Luijten, B. R. Hemenway, R. Grzybowski, and E. Schiattarella, "Designing a Crossbar Scheduler for HPC Applications," *IEEE Micro*, vol. 26, pp. 58–71, May-June 2006.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity, Data Center Network Architecture," in *Proceedings of ACM SIGCOMM*, August 2008.

[4] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers," in *Proceedings of ACM SIGCOMM*, Aug-Sep 2010.

[5] L. Tassiullas and A. Ephremides, "Stability Properties of Constrained Queuing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1936–1949, December 1992.

[6] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Transactions on Communications*, vol. 47, pp. 1260–1267, August 1999.

[7] N. Mckeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 188–201, April 1999.

[8] Y. Li, S. Panwar, and H. J. Chao, "On the Performance of a Dual Round-Robin Switch," in *Proc. of IEEE INFOCOM*, April 2001.

[9] T. E. Anderson, S. S. Owichi, J. B. Saxe, and C. P. Thacher, "High Speed Switch Scheduling for Local Area Networks," *ACM Transactions on Computer Systems*, vol. 11, pp. 319–352, November 1993.

[10] J. G. Dai and B. Prabhakar, "The Throughput of Data Switches with and without Speedup," in *Proceedings of IEEE INFOCOM*, (Tel Aviv, Israel), 2000.

[11] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch," in *Proc. of IEEE INFOCOM*, March 1999.

[12] R. Rojas-Cessa, E. Oki, and H. J. Chao, "On the Combined Input-Crosspoint Buffered Packet Switch with Round-Robin Arbitration," *IEEE Transactions on Communications*, vol. 53, pp. 1945–1951, November 2005.

[13] T. Javidi, R. Magill, and T. Hrabik, "A High Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric," in *Proceedings of IEEE ICC*, (Helsinki, Finland), June 2001.

[14] L. Mhamdi and M. Hamdi, "MCBF: a High-Performance Scheduling Algorithm for Buffered Crossbar Switches," *IEEE Communications Letters*, vol. 7, pp. 451–453, September 2003.

[15] S.-T. Chuang, S. Iyer, and N. McKeown, "Practial Algorithms for Performance Guarantees in Buffered Crossbars," in *Proceedings of IEEE INFOCOM*, (Miami, Florida), March 2005.

[16] J. Turner, "Strong Performance Guarantees for Asynchronous Crossbar Schedulers," in *Proceedings of IEEE INFOCOM*, (Spain), April 2006.

[17] P. Giaccone, E. Leonardi, and D. Shah, "On th emaximal throughput of networks with finite buffers and its application to buffered crossbars," in *Proceedings of IEEE Infocom*, (Miami, FL), March 2005.

[18] M. Bayati, B. Prabhakar, D. Shah, and M. Sharma, "Iterative Scheduling Algorithms," in *Proc. of IEEE INFOCOM*, May 2007.

[19] S. Rajagopalan, D. Shah, and J. Shin, "Network adiabatic theorem: an efficient randomized protocol for contention resolution," in *Proceedings of ACM SIGMETRICS*, June 2009.

[20] L. Jiang and J. Walrand, "A Distributed CSMA Algorithm for Throughput and Utility Maximization in Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 960–972, June 2010.

[21] J. Ni and R. Srikant, "Q-CSMA: Queue-Length Based CSMA/CA Algorithms for Achieving Maximum Throughput and Low Delay in Wireless Networks," *Proceedings of IEEE Infocom*, Apr. 2010.

[22] J. Liu, Y. Yi, A. Proutire, M. Chiang, and H. V. Poor, "Towards utility-optimal random access without message passing," *Wireless Communications and Mobile Computing*, vol. 10, pp. 115–128, Jan. 2010.

[23] L. Jiang, M. Leconte, J. Ni, R. Srikant, and J. Walrand, "Fast Mixing of Parallel Glauber Dynamics and Low-Delay CSMA Scheduling," *submitted*, August 2010.

[24] J. Ghaderi and R. Srikant, "On the Design of Efficient CSMA Algorithms for Wireless Networks," *submitted*, 2010.

[25] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

[26] S. Rajagopalan and D. Shah, "Aloha that works," *submitted*, Nov. 2008.

[27] Y. Shen, S. S. Panwar, and H. J. Chao, "Providing 100% Throughput in a Buffered Crossbar Switch," in *Proceedings of IEEE HPSR*, (Brooklyn, New York), May-June 2007.

[28] Y. Shen, S. S. Panwar, and H. J. Chao, "SQUID: A Practical 100Scheduler for Crosspoint Buffered Switches," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 1119–1131, August 2010.

[29] P. Giaccone, B. Prabhakar, and D. Shah, "Toward Simple, High Performance Schedulers for High-Aggregate Bandwidth Switches," in *Proceedings of IEEE INFOCOM*, (New York), 2002.

[30] F. Kelly, *Reversibility and Stochastic Networks*. Wiley, 1979.

[31] A. Eryilmaz, R. Srikant, and J. R. Perkins, "Stable Scheduling Policies for Fading Wireless Channels," *IEEE/ACM Transactions on Networking*, vol. 13, pp. 411–424, April 2005.

[32] M. Jerrum and A. Sinclair, "Approximating the permanent," *SIAM Journal of Computing*, vol. 18, pp. 1149–1178, 1989.

[33] P. Bremaud, *Markov chains, Gibbs fields, Monte Carlo simulation and queues*. Springer-Verlag, 2001.

APPENDIX

A. Proof of Lemma 3

Proof: The transition occurs only when the VOQs in $\mathbf{H}$ satisfy the conditions below:

1) For any $(i,j) \in \mathbf{X} \cap \overline{\mathbf{X}'}$: the VOQ is selected by $\mathbf{H}$ and decides to change its scheduling decision from 1 to 0, which happens with probability $\overline{p}_{ij}$.
2) For any $(k,l) \in \overline{\mathbf{X}} \cap \mathbf{X}'$: the VOQ is selected by $\mathbf{H}$ and decides to change its scheduling decision from 0 to 1, which happens with probability p_{kl}.
3) For any $(u,v) \in \mathbf{X} \cap \mathbf{X}' \cap \mathbf{H}$: the VOQ was in the DISQUO schedule of the previous time slot, and even though selected by $\mathbf{H}$ it decides to keep its state, which occurs with probability p_{uv}.
4) For any $(x,y) \in \mathbf{H} \cap \overline{\mathbf{X} \cup \mathbf{X}'} \cap \overline{\mathcal{N}(\mathbf{X})}$: neither the VOQ nor any of its neighbors was in the DISQUO schedule of the previous time slot, and though selected by $\mathbf{H}$ it decides to keep its schedule, which occurs with probability $\overline{p}_{xy}$. Since $\mathbf{H}$ is a DISQUO schedule and $\overline{\mathbf{X}} \cap \mathbf{X}' \in \mathbf{H}$, $\mathbf{H} \cap \mathcal{N}(\overline{\mathbf{X}} \cap \mathbf{X}') = \emptyset$. Thus $\mathbf{H} \cap \overline{\mathbf{X} \cup \mathbf{X}'} \cap \overline{\mathcal{N}(\mathbf{X})} = \mathbf{H} \cap \overline{\mathbf{X} \cup \mathbf{X}'} \cap \overline{\mathcal{N}(\mathbf{X} \cup \mathbf{X}')}$. We replace $\mathbf{H} \cap \overline{\mathbf{X} \cup \mathbf{X}'} \cap \overline{\mathcal{N}(\mathbf{X})}$ by $\mathbf{H} \cap \overline{\mathbf{X} \cup \mathbf{X}'} \cap \overline{\mathcal{N}(\mathbf{X} \cup \mathbf{X}')}$ in Eq. (15) for the proof of the stationary distribution in the following.

Since $\mathbf{H}$ is a permutation of the inputs and outputs, for any two VOQs in $\mathbf{H}$, they are not neighbors of each other. Therefore, they can make the scheduling decisions independently. We can then multiply the probabilities of all the four categories above, which leads to the transition probability given by Eq. (15). ■

B. Proof of Lemma 4

Proof: Suppose that $\mathbf{X}$ is a DISQUO schedule, and it has k non-zero elements: $(i_1,j_1), (i_2,j_2) \cdots (i_k,j_k) \in \mathbf{X}$. Let $\mathbf{X}_l$ represent a DISQUO schedule which has l non-zero elements: $(i_1,j_1), (i_2,j_2) \cdots (i_l,j_l) \in \mathbf{X}_l \subseteq \mathbf{X}$, $0 \le l \le k$. We can see that $\mathbf{X}_0 = \mathbf{0}$ and $\mathbf{X}_k = \mathbf{X}$. Since $\mathbf{X}$ is a DISQUO schedule, $\mathbf{X}_l$ is also a DISQUO schedule and $\mathbf{X}_{l-1} \cup \mathbf{X}_l = \mathbf{X}_l \in \mathcal{X}$. Therefore, the system can make a transition from $\mathbf{X}_{l-1}$ to $\mathbf{X}_l$ with positive probability when $(i_l,j_l) \in \mathbf{H}(n)$, as we already proved in Lemma 3. Hence, state $\mathbf{X}_0$ can reach any state $\mathbf{X} \in \mathcal{X}$ with positive probability in a finite number of steps and vice versa. Thus, the Markov chain is irreducible and positive recurrent. ■

C. Lemmas for System Stability

Lemma 8. *Suppose that $T(\cdot)$ is a function defined on a set $\mathcal{X}$. For any probability distribution μ on $\mathcal{X}$, define the function:*

$$F(\mu, T(\mathbf{X})) = E_\mu[T(\mathbf{X})] + \mathcal{H}(\mu), \tag{25}$$

where $\mathcal{H}(\mu)$ is the entropy function: $-\sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X})\log\mu(\mathbf{X})$. Then $F(\cdot)$ is uniquely maximized by the distribution:

$$\mu^*(\mathbf{X}) = \frac{1}{Z}\exp(T(\mathbf{X})), \tag{26}$$

where $Z = \sum_{\mathbf{X}\in\mathcal{X}}\exp(T(\mathbf{X}))$.

Proof: For any probability distribution μ, we have:

$$\begin{aligned}
& F(\mu, T(\mathbf{X})) \\
= \;& E_\mu[T(\mathbf{X})] + \mathcal{H}(\mu) \\
= \;& \sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X})T(\mathbf{X}) - \sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X})\log\mu(\mathbf{X}) \\
= \;& \sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X})(\log\mu^*(\mathbf{X}) + \log Z) - \sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X})\log\mu(\mathbf{X}) \\
= \;& \sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X})\log Z + \sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X})\log\frac{\mu^*(\mathbf{X})}{\mu(\mathbf{X})} \\
\leq \;& \log Z\sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X}) + \log\Big(\sum_{\mathbf{X}\in\mathcal{X}}\mu(\mathbf{X})\frac{\mu^*(\mathbf{X})}{\mu(\mathbf{X})}\Big) \\
= \;& \log Z,
\end{aligned} \tag{27}$$

with equality holding only when $\mu = \mu^*$. **QED** ■

Note that when $T(\mathbf{X}) = 0$, the uniform distribution maximizes $F(\mu, 0)$, and we have:

$$F(\mu, 0) = \mathcal{H}(\mu) \leq \log Z = \log|\mathcal{X}| \tag{28}$$

where $|\mathcal{X}|$ is the size of $\mathcal{X}$.

Lemma 9. *Let $W(\cdot)$ be the weight function and $W^*(\mathbf{X})$ the maximum weight. Define the set:*

$$\mathcal{K} = \{\mathbf{X}\in\mathcal{X} : W(\mathbf{X}) \leq (1-\epsilon)W^*(\mathbf{X})\}. \tag{29}$$

Then, we have:

$$\pi(\mathcal{K}) \leq \frac{\log|\mathcal{X}|}{\epsilon W^*(\mathbf{X})} \tag{30}$$

Proof: As shown in Eq. (16), for a schedule $\mathbf{X}\in\mathcal{X}$, its stationary distribution is: $\pi(\mathbf{X}) = \frac{1}{\mathcal{Z}}\prod_{(i,j)\in\mathbf{X}}e^{(w_{ij}(n))} = \frac{1}{\mathcal{Z}}e^{W(\mathbf{X})}$. According to Lemma 8, π maximizes $F(\mu, W(\mathbf{X}))$.

Let $\mathbf{X}^*$ be the schedule that maximizes the weight, and π' be the distribution that assigns all probability on $\mathbf{X}^*$ such that:

$$\pi'(\mathbf{X}) = \begin{cases} 1 & \text{if } \mathbf{X} = \mathbf{X}^* \\ 0 & \text{otherwise} \end{cases}$$

Then we have:

$$\begin{aligned}
F(\pi', W(\mathbf{X})) &= E_{\pi'}[W(\mathbf{X})] + \mathcal{H}(\pi') \\
&= W^*(\mathbf{X}) + \mathcal{H}(\pi') \\
&\leq F(\pi, W(\mathbf{X})) = E_\pi[W(\mathbf{X})] + \mathcal{H}(\pi) \\
&\leq W^*(\mathbf{X})(1-\pi(\mathcal{K})) \\
&\quad + W^*(\mathbf{X})(1-\epsilon)\pi(\mathcal{K}) + \mathcal{H}(\pi) \\
&= W^*(\mathbf{X})(1-\epsilon\pi(\mathcal{K})) + \mathcal{H}(\pi)
\end{aligned} \tag{31}$$

The last step in Eq. (31) uses Eq. (29). So,

$$\begin{aligned}
W^*(\mathbf{X}) + \mathcal{H}(\pi') &\leq W^*(\mathbf{X})(1-\epsilon\pi(\mathcal{K})) + \mathcal{H}(\pi) \\
\epsilon\pi(\mathcal{K})W^*(\mathbf{X}) &\leq \mathcal{H}(\pi) - \mathcal{H}(\pi') \leq \mathcal{H}(\pi) \leq \log|\mathcal{X}| \\
\pi(\mathcal{K}) &\leq \frac{\log|\mathcal{X}|}{\epsilon W^*(\mathbf{X})}
\end{aligned} \tag{32}$$

■

D. Proof of System Convergence

Before presenting the proof, we need to introduce some preliminaries. We will first define a *matrix norm*, which will be useful in determining the mixing time of a finite-state Markov chain.

Definition 7. *(Matrix norm) Consider a $|\mathbf{\Omega}|\times|\mathbf{\Omega}|$ non-negative matrix $\boldsymbol{A}\in\mathbb{R}_+^{|\mathbf{\Omega}|\times|\mathbf{\Omega}|}$ and a given vector $\boldsymbol{\mu}\in\mathbb{R}_+^{|\mathbf{\Omega}|}$. Then, the matrix norm of $\boldsymbol{A}$ with respect to $\boldsymbol{\mu}$ is defined as:*

$$\|\mathbf{A}\|_{\boldsymbol{\mu}} = \sup_{\boldsymbol{\nu}:E_{\boldsymbol{\mu}}[\boldsymbol{\nu}]=0}\frac{\|\mathbf{A}\boldsymbol{\nu}\|_{2,\boldsymbol{\mu}}}{\|\boldsymbol{\nu}\|_{2,\boldsymbol{\mu}}}, \tag{33}$$

where $\boldsymbol{\nu}\in\mathbb{R}_+^{|\mathbf{\Omega}|}$ and $E_{\boldsymbol{\mu}}[\boldsymbol{\nu}] = \sum_i\mu_i\nu_i$.

It is easy to check that the matrix norm has the following properties [26]:

Property 6. *For a matrix $\mathbf{A}\in\mathbb{R}_+^{|\mathbf{\Omega}|\times|\mathbf{\Omega}|}$, $\boldsymbol{\pi}\in\mathbb{R}_+^{|\mathbf{\Omega}|}$ and $a\in\mathbb{R}$:*

$$\|a\mathbf{A}\|_{\boldsymbol{\pi}} = |a|\|\mathbf{A}\|_{\boldsymbol{\pi}}. \tag{34}$$

Property 7. **A** *and* **B** *are the transition matrices of two reversible Markov chains. They have the same stationary distribution which is $\boldsymbol{\pi}$. We then have:*

$$\|\mathbf{AB}\|_{\boldsymbol{\pi}} \leq \|\mathbf{A}\|_{\boldsymbol{\pi}}\|\mathbf{B}\|_{\boldsymbol{\pi}}. \tag{35}$$

Property 8. *Let* **P** *be the transition matrix of a reversible Markov chain, which has the stationary distribution $\boldsymbol{\pi}$. We then have:*

$$\|\mathbf{P}\|_{\boldsymbol{\pi}} \leq e_{\max}, \tag{36}$$

where $e_{\max} = \max\{|e| : |e| \neq 1, e \text{ is an eigenvalue of } \mathbf{P}\}$ and $0 < e_{max} < 1$.

With the definition and these properties, it follows that for any distribution $\boldsymbol{\mu}$ on $\mathbf{\Omega}$, , we have [26]:

$$\Big\|\frac{\boldsymbol{\mu}\mathbf{P}}{\boldsymbol{\pi}} - 1\Big\|_{2,\boldsymbol{\pi}} \leq \|\mathbf{P}^*\|_{\boldsymbol{\pi}}\Big\|\frac{\boldsymbol{\mu}}{\boldsymbol{\pi}} - 1\Big\|_{2,\boldsymbol{\pi}}. \tag{37}$$

Then, if the Markov chain is time-reversible, we have:

$$\Big\|\frac{\boldsymbol{\mu}(\tau)}{\boldsymbol{\pi}} - 1\Big\|_{2,\boldsymbol{\pi}} \leq \|\mathbf{P}\|_{\boldsymbol{\pi}}^\tau\Big\|\frac{\boldsymbol{\mu}(0)}{\boldsymbol{\pi}} - 1\Big\|_{2,\boldsymbol{\pi}} \leq e_{max}^\tau\Big\|\frac{\boldsymbol{\mu}(0)}{\boldsymbol{\pi}} - 1\Big\|_{2,\boldsymbol{\pi}}. \tag{38}$$

Since

$$\begin{aligned}
\Big\|\frac{\boldsymbol{\mu}(0)}{\boldsymbol{\pi}} - 1\Big\|_{2,\boldsymbol{\pi}} &= \sqrt{\sum_{i\in\mathbf{\Omega}}\pi(i)\Big(\frac{\mu(i,0)}{\pi(i)} - 1\Big)^2} \\
&\leq \sqrt{\frac{1}{\min_i\pi(i)}},
\end{aligned} \tag{39}$$

for any $\delta > 0$, we have $\left\| \frac{\boldsymbol{\mu}(\tau)}{\boldsymbol{\pi}} - 1 \right\|_{2,\boldsymbol{\pi}} \leq \delta$ if

$$\tau \geq \frac{\frac{1}{2}\log 1/\pi_{min} + \log 1/\delta}{\log 1/e_{max}}, \tag{40}$$

where $\pi_{min} = \min_i \pi_i$. The equation above suggests that the mixing time of a reversible Markov chain with transition matrix $\mathbf{P}$ scales with $1 - e_{max}$, where $e_{max} = \max\{|e| \neq 1 : \text{e is an eigenvalue of } \mathbf{P}\}$. Therefore, in the following, we will refer to the mixing time of a reversible Markov chain with transition matrix $\mathbf{P}$ as: $T_{mix} = \frac{1}{1-e_{max}}$.

Recall that following the updating rules of DISQUO algorithm, there are at most N updates at every time slot, where N is the number of ports. Therefore, we will consider a multiple-update Glauber dynamics defined as follows.

Definition 8. *(Multiple-update Glauber dynamics) Consider a graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$, with $\mathbf{W} = [W_i]_{i\in\mathbf{V}}$, which is a vector of weights associated with the vertices. Multiple update Glauber dynamics (MUGD) is a Markov chain over $\mathcal{I}(\mathbf{G})$. Suppose that the chain is at state $\mathbf{X}(n-1) = [X_i(n-1)]_{i\in\mathbf{V}}$ at time $n-1$. The next transition of multiple-update Glauber dynamics follows the rules:*

- *Randomly pick a set $\mathbf{H}(n) \in \mathcal{I}(G)$ at random.*
- *For $i \in \mathbf{H}(n)$:*
  - *If $\forall j \in \mathcal{N}(i)$, $X_j(n-1) = 0$, then*
$$X_i(n) = \begin{cases} 1 & \text{with probability } \frac{\exp(W_i)}{1+\exp(W_i)} \\ 0 & \text{otherwise.} \end{cases}$$
  - *Otherwise, $X_i(n) = 0$.*
- *$X_i(n) = X_i(n-1)$, for all $i \notin \mathbf{H}(n)$.*

The transition matrix is similar to Eq. (15) but with a vector of fixed weights. The multiple-update Glauber dynamics is also a positive recurrent, time-reversible Markov chain. It is easy to verify that the product-form stationary distribution in Eq. (1) satisfies the detailed balance equation in Eq. (18) that it is also the stationary distribution of the multiple-update Glauber dynamics. In the following lemma, we will give an upper bound on the mixing time of the multiple-update Glauber dynamics.

Lemma 10. *(Mixing time of multiple-update Glauber dynamics) Let $\mathbf{P}$ be the transition matrix of the multiple-update Glauber dynamics on a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, for which there are N vertices with weights $\mathbf{W} = [W_i]_{i\in\mathbf{V}}$. We have:*

$$T_{mix} \leq 2^{6N}\exp(4NW_{max}), \tag{41}$$

where $W_{max} = \max_{i\in\mathbf{V}} W_i$.

Proof: For a nonempty set $\mathbf{A} \subset \mathcal{I}(G)$, we have:

$$\pi(\mathbf{A}) = \sum_{i\in\mathbf{A}} \pi(i).$$

Let us define the following:

$$F(\mathbf{A}) = \sum_{i\in\mathbf{A}, j\in\mathbf{A}^c} \pi(i)p_{ij}.$$

The conductance of the transition matrix $\mathbf{P}$ is defined as:

$$\phi(\mathbf{P}) = \min_{\mathbf{A}\subset\mathcal{I}(G):\pi(\mathbf{A})\leq\frac{1}{2}} \frac{F(\mathbf{A})}{\pi(\mathbf{A})}.$$

There is a well-known conductance bound [32], [33] with the form:

$$e_{max} \leq 1 - \frac{\phi^2(\mathbf{P})}{2}.$$

Now, we have:

$$\begin{aligned}
\phi(\mathbf{P}) &= \min_{\mathbf{A}\subset\mathcal{I}(G):\pi(\mathbf{A})\leq\frac{1}{2}} \frac{F(\mathbf{A})}{\pi(\mathbf{A})} \\
&= \min_{\mathbf{A}\subset\mathcal{I}(G):\pi(\mathbf{A})\leq\frac{1}{2}} \frac{\sum_{\mathbf{X}\in\mathbf{A},\mathbf{X}'\in\mathbf{A}^c}\pi(\mathbf{X})P(\mathbf{X},\mathbf{X}')}{\pi(\mathbf{A})} \\
&\geq 2\min_{\mathbf{A}\in\mathcal{I}(G)} P(\mathbf{A},\mathbf{A}^c) \\
&\geq 2\min_{P(\mathbf{X},\mathbf{X}')\neq 0} \pi(\mathbf{X})P(\mathbf{X},\mathbf{X}') \\
&\geq 2\min_{\mathbf{X}}\pi(\mathbf{X}) \min_{\mathbf{X}\neq\mathbf{X}', P(\mathbf{X},\mathbf{X}')\neq 0} P(\mathbf{X},\mathbf{X}')
\end{aligned}$$

For the Glauber dynamics, the stationary distribution can be lower bounded by:

$$\begin{aligned}
\pi(\mathbf{X}) &\geq \frac{1}{\sum_{\mathbf{X}\in\mathcal{I}(G)}\exp(\sum_{i\in\mathbf{X}} W_i)} \\
&\geq \frac{1}{|\mathcal{I}(G)|\exp(NW_{max})} \\
&\geq \frac{1}{2^N\exp(NW_{max})}
\end{aligned}$$

Also, we have:

$$P(\mathbf{X},\mathbf{X}') \geq \frac{1}{2^N}\Big(\frac{1}{1+\exp(W_{max})}\Big)^N.$$

So,

$$\begin{aligned}
\phi(\mathbf{P}) &\geq \frac{2}{2^{2N}(1+\exp(W_{max}))^N\exp(NW_{max})} \\
&\geq \frac{2}{2^{3N}\exp(2NW_{max})}
\end{aligned}$$

Thus,

$$e_{max} \leq 1 - \frac{2}{2^{6N}\exp(4NW_{max})} \leq 1 - \frac{1}{2^{6N}\exp(4NW_{max})}.$$

Since $T_{mix} = \frac{1}{1-e_{max}}$, we have:

$$T_{mix} \leq 2^{6N}\exp(4NW_{max}).$$

■

Now, we are ready to prove Lemma 6. We will first identify the condition for the system to converge in Lemma 11. Then, in Lemma 12, we will prove that if the weight function $f(\cdot)$ are well designed, the condition for the system convergence can be satisfied, and thus finish the proof of Lemma 6. The proof of Lemma 11 is mainly adapted from Ref. [19], [26].

Let $\mathbf{P}_n$ denote the transition matrix at time n. $e_{max}(n) = \max\{|e| : |e| \neq 1, \text{e is the eigenvalue of } \mathbf{P}_n\}$, and $T_n = \frac{1}{1-e_{max}(n)}$, which is the mixing time of the multiple-update Glauber dynamics with weight vector $\mathbf{W}(n)$.

In the following Lemma, we will prove that given the condition that $\alpha_n T_{n+1} \leq \delta/8$ $(\forall \delta > 0)$, the system can

converge within finite time, where α_n is defined as Eq. (42). We will also give an upper bound on the mixing time of the system.

Lemma 11. *If $\alpha_n T_{n+1} \leq \delta/8$, then for any $\delta > 0$, $\|\boldsymbol{\mu}_n - \boldsymbol{\pi}_n\|_{TV} \leq \delta$ holds for all $n \geq n*$, where T_{n+1} is the mixing time of the multiple-update Glauber dynamics with weight vector $\mathbf{W}(n+1)$,*

$$\alpha_n = \sum_{i,j} f'(\tilde{Q}_{ij}(n)) + f'(\tilde{Q}_{ij}(n+1)), \tag{42}$$

and

$$n^* = \min_n \sum_{i=1}^{n} \frac{1}{T_i^2} \geq \log(\frac{2}{\delta}) + \frac{N^2}{2}\Big(\log 2 + W_{max}(0)\Big)., \tag{43}$$

Proof: The stationary distributions for the multiple-update Glauber dynamics with weight vectors $\mathbf{W}(n)$ and $\mathbf{W}(n+1)$ can be written as:

$$\boldsymbol{\pi}_n(\mathbf{X}) = \frac{1}{Z_n} \exp(\sum_{(i,j)\in\mathbf{X}} W_{ij}(n)),$$

and

$$\boldsymbol{\pi}_{n+1}(\mathbf{X}) = \frac{1}{Z_{n+1}} \exp(\sum_{(i,j)\in\mathbf{X}} W_{ij}(n+1)),$$

respectively. So,

$$\frac{\boldsymbol{\pi}_{n+1}(\mathbf{X})}{\boldsymbol{\pi}_n(\mathbf{X})} = \frac{Z_n}{Z_{n+1}} \exp\Big(\sum_{(i,j)\in\mathbf{X}} (W_{ij}(n+1) - W_{ij}(n))\Big), \tag{44}$$

and

$$\begin{aligned} \frac{Z_n}{Z_{n+1}} &\leq \frac{\sum_{\mathbf{X}\in\mathcal{I}(\mathbf{G})} \exp(\sum_{(i,j)\in\mathbf{X}} W_{ij}(n))}{\sum_{\mathbf{X}\in\mathcal{I}(\mathbf{G})} \exp(\sum_{(i,j)\in\mathbf{X}} W_{ij}(n+1))} \\ &\leq \max_{\mathbf{X}} \exp(\sum_{(i,j)\in\mathbf{X}} W_{ij}(n) - W_{ij}(n+1)). \end{aligned}$$

Note that $W_{ij}(n) = f(\tilde{Q}_{ij}(n))$, and $f(\cdot)$ is a increasing concave function that $f(b) - f(a) \leq f'(a)(b-a)$. Therefore,

$$\begin{aligned} W_{ij}(n) - W_{ij}(n+1) &\leq f'(\tilde{Q}_{ij}(n+1))(\tilde{Q}_{ij}(n) - \tilde{Q}_{ij}(n+1)) \\ &\leq f'(\tilde{Q}_{ij}(n+1)). \end{aligned}$$

The equation above is according to the fact that at every time slot, there is at most one arrival and one departure that we have $-1 \leq \tilde{Q}_{ij}(n) - \tilde{Q}_{ij}(n+1) \leq 1$. So,

$$\begin{aligned} \frac{Z_n}{Z_{n+1}} &\leq \max_{\mathbf{X}} \exp(\sum_{(i,j)\in\mathbf{X}} f'(\tilde{Q}_{ij}(n+1))) \\ &\leq \exp(\sum_{i,j} f'(\tilde{Q}_{ij}(n+1))) \end{aligned} \tag{45}$$

Similarly, we have

$$\frac{Z_{n+1}}{Z_n} \leq \exp(\sum_{i,j} f'(\tilde{Q}_{ij}(n))) \tag{46}$$

From Eq. (44) and Eq. (45), we have:

$$\begin{aligned} \frac{\boldsymbol{\pi}_{n+1}(\mathbf{X})}{\boldsymbol{\pi}_n(\mathbf{X})} &= \frac{Z_n}{Z_{n+1}} \exp\Big(\sum_{(i,j)\in\mathbf{X}} (W_{ij}(n+1) - W_{ij}(n))\Big) \\ &\leq \exp(\sum_{i,j} f'(\tilde{Q}_{ij}(n)) + f'(\tilde{Q}_{ij}(n+1))), \end{aligned} \tag{47}$$

and also,

$$\frac{\boldsymbol{\pi}_n(\mathbf{X})}{\boldsymbol{\pi}_{n+1}(\mathbf{X})} \leq \exp(\sum_{i,j} f'(\tilde{Q}_{ij}(n)) + f'(\tilde{Q}_{ij}(n+1))). \tag{48}$$

Define $\alpha_n = \sum_{i,j}\Big[f'(\tilde{Q}_{ij}(n)) + f'(\tilde{Q}_{ij}(n+1))\Big]$. We have:

$$\exp(-\alpha_n) \leq \frac{\boldsymbol{\pi}_n(\mathbf{X})}{\boldsymbol{\pi}_{n+1}(\mathbf{X})} \leq \exp(\alpha_n) \tag{49}$$

Recall that $\alpha_n T_{n+1} \leq \frac{\delta}{8}$, and $T_{n+1} = \frac{1}{1-e_{max}(n+1)} \geq 1$ is the mixing time of the multiple-update Glauber dynamics with weight vector $\mathbf{W}(n)$. Since δ is any small positive number, we have $0 < \alpha_n < 1$. Since $1 - x \leq e^{-x}$ and $e^x \leq 1 + 2x$ for all $x \in [0,1]$, we have

$$-\alpha_n \leq \frac{\boldsymbol{\pi}_n(\mathbf{X})}{\boldsymbol{\pi}_{n+1}(\mathbf{X})} - 1 \leq 2\alpha_n.$$

So,

$$\Big(\frac{\boldsymbol{\pi}_n(\mathbf{X})}{\boldsymbol{\pi}_{n+1}(\mathbf{X})} - 1\Big)^2 \leq 4\alpha_n^2.$$

Then,

$$\begin{aligned} \|\boldsymbol{\pi}_{n+1} - \boldsymbol{\pi}_n\|_{2,1/\boldsymbol{\pi}_{n+1}}^2 &= \|\frac{\boldsymbol{\pi}_n}{\boldsymbol{\pi}_{n+1}} - 1\|_{2,\boldsymbol{\pi}_{n+1}}^2 \\ &= \sum_{\mathbf{X}} \pi_{n+1}(\mathbf{X})\Big(\frac{\pi_n(\mathbf{X})}{\pi_{n+1}(\mathbf{X})} - 1\Big)^2 \\ &\leq 4\alpha_n^2 \sum_{\mathbf{X}} \pi_{n+1}(\mathbf{X}) = 4\alpha_n^2 \end{aligned}$$

Thus,

$$\|\boldsymbol{\pi}_{n+1} - \boldsymbol{\pi}_n\|_{2,1/\boldsymbol{\pi}_{n+1}} \leq 2\alpha_n.$$

The distance between $\boldsymbol{\mu}_n$ and $\boldsymbol{\pi}_n$ can then be bounded by:

$$\begin{aligned} \|\frac{\boldsymbol{\mu}_n}{\boldsymbol{\pi}_n} - 1\|_{2,\boldsymbol{\pi}_n} &= \|\boldsymbol{\mu}_n - \boldsymbol{\pi}_n\|_{2,1/\boldsymbol{\pi}_n} \\ &\leq \|\boldsymbol{\mu}_n - \boldsymbol{\pi}_{n-1}\|_{2,1/\boldsymbol{\pi}_n} \\ &\quad + \|\boldsymbol{\pi}_{n-1} - \boldsymbol{\pi}_n\|_{2,1/\boldsymbol{\pi}_n} \\ &\leq \|\boldsymbol{\mu}_n - \boldsymbol{\pi}_{n-1}\|_{2,1/\boldsymbol{\pi}_n} + 2\alpha_{n-1}. \end{aligned} \tag{50}$$

Note that

$$\begin{aligned} \|\boldsymbol{\mu}_n - \boldsymbol{\pi}_{n-1}\|_{2,1/\boldsymbol{\pi}_n}^2 &= \sum_{\mathbf{X}} \frac{1}{\pi_n(\mathbf{X})}(\mu_n(\mathbf{X}) - \pi_{n-1}(\mathbf{X}))^2 \\ &= \sum_{\mathbf{X}} \frac{\pi_{n-1}(\mathbf{X})}{\pi_n(\mathbf{X})} \frac{1}{\pi_{n-1}(\mathbf{X})} \\ &\quad \cdot(\mu_n(\mathbf{X}) - \pi_{n-1}(\mathbf{X}))^2 \\ &\leq e^{(\alpha_{n-1})} \|\boldsymbol{\mu}_n - \boldsymbol{\pi}_{n-1}\|_{2,1/\boldsymbol{\pi}_{n-1}}^2 \end{aligned} \tag{51}$$

From Eq. (50) and (51), we have:

$$\begin{aligned}\|\frac{\boldsymbol{\mu}_n}{\boldsymbol{\pi}_n}-1\|_{2,\boldsymbol{\pi}_n} &\leq \exp(\alpha_{n-1}/2)\|\boldsymbol{\mu}_n-\boldsymbol{\pi}_{n-1}\|_{2,1/\boldsymbol{\pi}_{n-1}} \\ &\quad +2\alpha_{n-1} \\ &\leq (1+\alpha_{n-1})\|\boldsymbol{\mu}_n-\boldsymbol{\pi}_{n-1}\|_{2,1/\boldsymbol{\pi}_{n-1}} \\ &\quad +2\alpha_{n-1} \qquad (52)\end{aligned}$$

Let us define

$$\beta_n = \|\boldsymbol{\mu}_{n+1}-\boldsymbol{\pi}_n\|_{2,1/\boldsymbol{\pi}_n}. \qquad (53)$$

Note that $\alpha_n \leq \alpha_n T_{n+1} \leq \delta/8$. If $\beta_n \leq \delta/2$, then from Eq. (52), we have:

$$\|\frac{\boldsymbol{\mu}_n}{\boldsymbol{\pi}_n}-1\|_{2,\boldsymbol{\pi}_n} \leq \delta, \qquad (54)$$

for all $n > n^*$. Therefore, to establish the result, we then have to prove that $\beta_n \leq \delta/2$ holds for any $n > n^*$. Consider the following equation:

$$\begin{aligned}\beta_{n+1} &= \|\boldsymbol{\mu}_{n+2}-\boldsymbol{\pi}_{n+1}\|_{2,1/\boldsymbol{\pi}_{n+1}} \\ &= \|\frac{\boldsymbol{\mu}_{n+2}}{\boldsymbol{\pi}_{n+1}}-1\|_{2,\boldsymbol{\pi}_{n+1}} \\ &= \|\frac{\boldsymbol{\mu}_{n+1}\mathbf{P}_{n+1}}{\boldsymbol{\pi}_{n+1}}-1\|_{2,\boldsymbol{\pi}_{n+1}} \\ &\leq \|\mathbf{P}_{n+1}\|_{\boldsymbol{\pi}_{n+1}}\|\boldsymbol{\mu}_{n+1}-\boldsymbol{\pi}_{n+1}\|_{2,1/\boldsymbol{\pi}_{n+1}} \\ &\leq e_{max}(n+1)\|\boldsymbol{\mu}_{n+1}-\boldsymbol{\pi}_{n+1}\|_{2,1/\boldsymbol{\pi}_{n+1}} \\ &\leq (1-\frac{1}{T_{n+1}}) \\ &\quad \cdot\Big(\|\boldsymbol{\mu}_{n+1}-\boldsymbol{\pi}_n\|_{2,\frac{1}{\boldsymbol{\pi}_{n+1}}}+\|\boldsymbol{\pi}_n-\boldsymbol{\pi}_{n+1}\|_{2,\frac{1}{\boldsymbol{\pi}_{n+1}}}\Big) \\ &\leq (1-\frac{1}{T_{n+1}})\Big(\|\boldsymbol{\mu}_{n+1}-\boldsymbol{\pi}_n\|_{2,1/\boldsymbol{\pi}_{n+1}}+2\alpha_n\Big) \\ &\leq (1-\frac{1}{T_{n+1}}) \\ &\quad \cdot\Big(\exp(\alpha_n/2)\|\boldsymbol{\mu}_{n+1}-\boldsymbol{\pi}_n\|_{2,1/\boldsymbol{\pi}_n}+2\alpha_n\Big) \\ &= (1-\frac{1}{T_{n+1}})\Big(\exp(\alpha_n/2)\beta_n+2\alpha_n\Big) \\ &\leq (1-\frac{1}{T_{n+1}})\Big((1+\alpha_n)\beta_n+2\alpha_n\Big) \qquad (55)\end{aligned}$$

Suppose that $\beta_n \leq \delta/2$, Eq. (55) can be written as:

$$\begin{aligned}\beta_{n+1} &\leq (1-\frac{1}{T_{n+1}})\Big(\frac{\delta}{2}+(2+\frac{\delta}{2})\alpha_n\Big) \\ &\leq (1-\frac{1}{T_{n+1}})\Big(\frac{\delta}{2}+(2+\frac{\delta}{2})\frac{\delta}{8T_{n+1}}\Big) \\ &\leq \frac{\delta}{2}-\frac{1}{T_{n+1}}\Big(\frac{\delta}{2}+(2+\frac{\delta}{2})\frac{\delta}{8T_{n+1}}-(2+\frac{\delta}{2})\frac{\delta}{8}\Big) \\ &\leq \frac{\delta}{2}. \qquad (56)\end{aligned}$$

From Eq. (56), we can see that, if $\beta_{n^*} \leq \delta/2$, then for any $n > n^*$, $\beta_n \leq \delta/2$.

In the following, we will find n^* which is the smallest number to satisfy $\beta_n \leq \delta/2$. Note that if $\beta_{n^*} \leq \delta/2$, then for any $n > n^*$, $\beta_n \leq \delta/2$. Therefore, for any $n < n^*$, $\beta_n > \delta/2$. So, for $n < n^*$, we have

$$\begin{aligned}\beta_n &\leq (1-\frac{1}{T_n})\Big((1+\alpha_{n-1})\beta_{n-1}+2\alpha_{n-1}\Big) \\ &\leq (1-\frac{1}{T_n})\Big((1+\alpha_{n-1})\beta_{n-1}+4\alpha_{n-1}\frac{\beta_{n-1}}{\delta}\Big) \\ &\leq (1-\frac{1}{T_n})\Big(1+(1+\frac{4}{\delta})\alpha_{n-1}\Big)\beta_{n-1} \\ &\leq (1-\frac{1}{T_n})\Big(1+\frac{1}{T_n}\Big)\beta_{n-1} \\ &= (1-\frac{1}{T_n^2})\beta_{n-1} \\ &\leq \exp(-\frac{1}{T_n^2})\beta_{n-1} \\ &\leq \exp(-\sum_{i=1}^{n}\frac{1}{T_i^2})\beta_0, \qquad (57)\end{aligned}$$

where β_0 can be written as:

$$\begin{aligned}\beta_0 &= \|\frac{\boldsymbol{\mu}_1}{\boldsymbol{\pi}_0}-1\|_{2,\boldsymbol{\pi}_0} \\ &= \|\boldsymbol{\mu}_0\mathbf{P}_0-\boldsymbol{\pi}_0\|_{2,1/\boldsymbol{\pi}_0} \\ &= e_{max}(0)\|\boldsymbol{\mu}_0-\boldsymbol{\pi}_0\|_{2,1/\boldsymbol{\pi}_0} \\ &\leq \sqrt{\frac{1}{\pi_{min}(0)}}, \qquad (58)\end{aligned}$$

where $\pi_{min}(0) = \min_i \pi_0(i) \geq \frac{1}{Z(0)} > \frac{1}{2^{N^2}\exp(N^2 W_{max}(0))}$. So,

$$\beta_0 \leq \Big(2\exp(W_{max}(0))\Big)^{N^2/2}. \qquad (59)$$

$\beta_{n^*} \leq \delta/2$ that it satisfies the condition:

$$\Big(2\exp(W_{max}(0))\Big)^{N^2/2}\exp(-\sum_{i=1}^{n^*}\frac{1}{T_i^2}) \leq \delta/2, \qquad (60)$$

or,

$$\sum_{i=1}^{n^*}\frac{1}{T_i^2} \geq \log(\frac{2}{\delta})+\frac{N^2}{2}\Big(\log 2+W_{max}(0)\Big). \qquad (61)$$

Note that T_i is bounded such that there always exists a n^* which can satisfies the condition above. ■

Lemma 12. *If $f(x) = \frac{\log(1+x)}{g(x)}$, there exists a constant C that when $\|\mathbf{Q}\| > C$, for any $\delta > 0$, $\alpha_n T_{n+1} \leq \delta/8$, where $g(x)$ is a function that satisfies the following conditions:*

- *$g(x) \geq 1$, for all $x \geq 0$.*
- *$g'(x) \geq 0$, for all $x \geq 0$.*
- *$\lim_{x\to\infty} g(f^{-1}(x)) = \infty$.*

Proof: We have:

$$f'(x) = \frac{1}{(1+x)g(x)}-\frac{\log(1+x)g'(x)}{g^2(x)} \leq \frac{1}{1+x}. \qquad (62)$$

Also,

$$f^{-1}(x) = \exp(xg(f^{-1}(x)))-1. \qquad (63)$$

Recall that

$$\begin{aligned}\alpha_n &= \sum_{i,j} f'(\tilde{Q}_{ij}(n)) + f'(\tilde{Q}_{ij}(n+1)) \\ &\leq N^2(f'(\tilde{Q}_{min}(n) + f'(\tilde{Q}_{min}(n+1)), \quad (64)\end{aligned}$$

where $\tilde{Q}_{min}(n) = \min_{ij} \tilde{Q}_{ij}(n)$, and

$$T_{n+1} \leq 2^{6N^2} \exp(4N^2 W_{max}(n+1)) \quad (65)$$

So,

$$\begin{aligned}\alpha_n T_{n+1} &\leq 2^{6N^2} \exp(4N^2 W_{max}(n+1)) \\ &\quad \cdot N^2(f'(\tilde{Q}_{min}(n) + f'(\tilde{Q}_{min}(n+1)) \\ &\leq N^2 2^{6N^2} \exp(4N^2 W_{max}(n+1)) \\ &\quad \cdot(\frac{1}{1+\tilde{Q}_{min}(n)} + \frac{1}{1+\tilde{Q}_{min}(n+1)}) \\ &\leq 2N^2 2^{6N^2} \exp(4N^2 W_{max}(n+1)) \\ &\quad \cdot(\frac{1}{\tilde{Q}_{min}(n+1)}) \\ &\leq 2N^2 2^{6N^2} \exp\left(4N^2(\frac{2N^2}{\epsilon})W_{min}(n+1)\right) \\ &\quad \cdot\frac{1}{f^{-1}(W_{min}(n+1))} \\ &= 2N^2 2^{6N^2} \exp\left(8N^4/\epsilon W_{min}(n+1)\right) \\ &\quad \cdot\frac{1}{\exp\left(W_{min}(n+1)g(f^{-1}(W_{min}(n+1)))\right) - 1}.\end{aligned}$$

Then,

$$\begin{aligned}\alpha_n T_{n+1} &\leq \exp\Big[\Big(8N^4/\epsilon \\ &\quad - g(f^{-1}(W_{min}(n+1)))\Big)W_{min}(n+1)\Big] \\ &\quad \cdot 2N^2 2^{6N^2}\Big(1 + \frac{1}{f^{-1}(W_{min}(n+1))}\Big) \quad (66)\end{aligned}$$

If $W_{max} \to \infty$, $W_{min} \to \infty$ such that $g(f^{-1}(W_{min}(n+1))) \to \infty$, and thus the value of $8N^4/\epsilon - g(f^{-1}(W_{min}(n+1))) \to \infty$. Therefore, for any $\delta > 0$, there exists a constant C such that when $\|\mathbf{Q}\| \geq C$, $\alpha_n T_{n+1} \leq \delta/8$ holds.

By proving Lemma 12, we give the sufficient condition that the system can converge, as stated in Lemma 11. Therefore, following the randomized scheduling algorithm, the inhomogeneous Markov chain can still converge to a stationary distribution, which can be expressed as Eq. (16). We finish the proof of Lemma 6. ■